Fast, High-Quality Two-Arm Rearrangement in Synchronous, Monotone Tabletop Setups

Rahul Shome[1] Kiril Solovey[2] Jingjin Yu[3] Kostas Bekris[3] Dan Halperin[4]

***Abstract*—Rearranging objects on a planar surface arises in a variety of robotic applications, such as product packaging. Using two arms can improve efficiency but introduces new computational challenges. This paper studies the problem structure of object rearrangement using two arms in synchronous, monotone tabletop setups and develops an optimal mixed integer model. It then describes an efficient and scalable algorithm, which first minimizes the cost of object transfers and then of moves between objects. This is motivated by the fact that, asymptotically, object transfers dominate the cost of solutions. Moreover, a lazy strategy minimizes the number of motion planning calls and results in significant speedups. Theoretical arguments support the benefits of using two arms and indicate that synchronous execution, in which the two arms perform together either transfers or moves, introduces only a small overhead. Experiments support these claims and show that the scalable method can quickly compute solutions close to the optimal for the considered setup.**

***Note to Practitioners*—Monotone tabletop rearrangement challenges arise in a variety of automation scenarios including product sorting or packing. Performing this task with two robotic manipulators introduces the overhead of coordinating them in the shared workspace, as well as an increase in the size of the underling search space. The objective of this work is to study the feasibility of such dual-arm solutions, providing both theoretical bounds, as well as a fast, and approximate solution. The approach leverages an effective algorithmic decomposition of the problem so as to take advantage of efficient motion planners and mixed-integer linear programming solvers. The proposed solution has been evaluated in settings that include delta-robots as well as *7-dof* manipulators. Interesting extensions of this work correspond to studying the case of additional arms, non-monotone and general manipulation scenarios.**

I. Introduction

AUTOMATION tasks in industrial and service robotics, such as product packing or sorting, often require sets of objects to be arranged in specific poses on a planar surface. Efficient and high-quality single-arm solutions have been proposed for such setups [1]. The proliferation of robot arms, however, including two-arm setups, implies that industrial settings can utilize multiple robots in the same workspace (Fig 1). This work explores a) the benefits and trade-offs of using two arms for solving object rearrangement instead of a single-arm, b) the combinatorial challenges involved and c) computationally efficient, high-quality and scalable methods.

[1]Rice University, TX, USA, [2]Stanford University, CA, USA, [3]Rutgers University, NJ, USA, and [4]Tel Aviv University, Israel.

Work by *RS*, *JU* and *KB* was supported by NSF CNS-1330789, IIS-1617744 and IIS-1734419. Work by *DH* was supported in part by the Israel Science Foundation (grant no. 1736/19), by NSF/US-Israel-BSF (grant no. 2019754), by the Israel Ministry of Science and Technology (grant no. 103129), by the Blavatnik Computer Science Research Fund, and by a grant from Yandex.

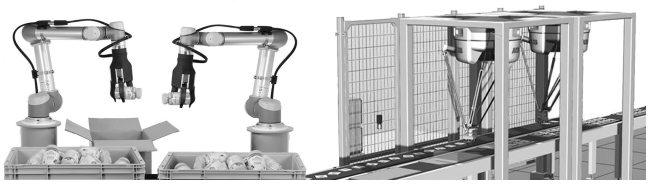


Fig. 1. Two-arm setups that can utilize algorithms proposed in this work.

A major motivation for this work is that the coordinated use of multiple arms to move multiple objects at the same time can result in significant improvements in efficiency of object rearrangement solutions. This arises from the following argument.

Observation 1: There are classes of tabletop rearrangement problems, where a k-arm ($k \geq 2$) solution can be arbitrarily better than the optimal single-arm one.

For instance, assume two arms that have full (overhand) access to a unit square planar tabletop. There are n objects on the table, divided into two groups of $\frac{n}{2}$ each. Objects in each group are ε-close to each other and to their goals. Let the distance between the two groups be on the order of 1, i.e., the two groups are at opposite ends of the table. The initial position of each end-effector is ε-close to a group of objects. Let the cost of each pick and drop action be c_{pd}, while moving the end-effector costs c_t per unit distance. Then, the 2-arm cost is no more than $2nc_{pd} + 2n\varepsilon c_t$. A single arm solution costs at least $2nc_{pd}+(2n-1)\varepsilon c_t+c_t$. If c_{pd} and ε are sufficiently small, the 2-arm cost can be arbitrarily better than the single-arm one. The argument also extends to k-arms relative to $(k-1)$-arms.

In most practical setups, the expectation is that a 2-arm solution will be close to half the cost (e.g., time-wise) of the single-arm case, which is a desirable improvement. While there is coordination overhead, the best 2-arm solution cannot do worse; simply let one of the arms carry out the best single arm solution while the other remains outside the workspace. This means whenever there is enough available space for k arms, it can be said:

Observation 2: For any rearrangement problem, the best k-arm ($k \geq 2$) solution cannot be worse than an optimal single arm solution.

The above observations motivate the development of scalable algorithmic tools for such two-arm rearrangement instances. This work considers certain relaxations to achieve this objective. In particular, monotone tabletop instances are considered, where the start and goal object poses do not overlap. Furthermore, the focus is on synchronized execution of pick-and-place actions by the two arms. i.e., where the two arms simultaneously transfer (different) objects, or simulta-

neously move towards picking the next objects. Theoretical arguments and experimental evaluation indicate that this does not significantly degrade solution quality.

It should be noted here that the solutions presented in this paper go beyond purely reasoning about task allocation and scheduling of objects to robots. The two-arm object rearrangement solution needs to consider geometric interactions and collisions between robots and obstacles through coordinated motion planning. Information about these real-world interactions need to be considered to yield solutions that are ultimately valid and collision-free.

The first contribution is the study of the combinatorial structure of synchronous, monotone two-arm rearrangement. Then, a mixed integer linear programming (`MILP`) model is proposed that achieves optimal coordinated solutions in this domain. The proposed efficient algorithm `Tour Over Matching` (`TOM`) significantly improves in scalability. `TOM` first optimizes the cost of object transfers and assigns objects to the two arms by solving an optimal matching problem. It minimizes the cost of moves from an object's goal to another object's start pose per arm as a secondary objective by employing a `TSP` solution. Most of the computation time is spent on the many calls to an underlying motion planner that coordinates the two arms. A lazy evaluation strategy is proposed, where motion plans are evaluated and validated for candidate solution sequences, as opposed to performing this expensive motion plan computation for the entire search space. This results in significant computational improvement and reduces the calls to the motion planner. An analysis studies the expected improvement in solution quality versus the single arm case, as well as the expected cost overhead from a synchronous solution.

Finally, experiments for i) a simple planar picker setting, and ii) two 7-DOF Kuka arms, demonstrate a nearly two-fold improvement against the single arm case for the proposed approach in practice. The algorithm exhibits close to optimal solutions and good scalability. The lazy evaluation strategy significantly improves computation costs for both the optimal and proposed methods.

The current archival version is an extension of previous work [2]. In particular, several additions have been made to provide a more holistic study of the problem:

1) A detailed illustration and expository text is provided for describing the algorithmic steps, especially regarding Section V.
2) Further analysis has been performed to study the problem (Section VII). Specifically, the result describing the asymptotic cost bound for asynchronous operation has been extended to the k-arm case. Additionally, a more detailed proof is provided for the asymptotic cost bound estimate in the case of synchronized two-arm operation.
3) The solutions from the benchmarks were post-processed so as to minimize delays introduced due to the synchronization assumption, and the data from this *smoothing* operation is provided in Section VIII-C.
4) A benchmark (Section VIII-D) has been included to demonstrate the efficacy of the proposed approach in problem domains where initial or target poses of objects do not lie in the shared workspace.

II. Related Work

The current work dealing with two-arm object rearrangement touches upon the challenging intersection of a variety of rich bodies of prior work. It is closely related to multi-robot planning and coordination where a challenge is the high dimensionality of the configuration space. Optimal strategies were developed for simpler instances of the problem [3], although in general the problem is known to be computationally hard [4]. Decentralized approaches [5] also used velocity tuning [6]. General multi-robot planning tries to plan for multiple high-dimensional platforms [7], [8] using sampling-based techniques. Recent advances provide scalable [9] and asymptotically optimal [10], [11] sampling based frameworks.

In some cases, by restricting the input of the problem to a certain type, it is possible to cast known hard instances of a problem as related algorithmic problems which have efficient solvers. For instance, unlabeled multi-robot motion planning can be reduced to pebble motion on graphs [12]; pebble motion can be reduced to network flow [13]; and single-arm object rearrangement can be cast as a traveling salesman problem [1]. These provide the inspiration to closely inspect the structure of the problem to derive efficient solutions.

In this work we leverage a connection between two-arm rearrangement and two combinatorial problems: (1) optimal matching [14] and (2) TSP. On the surface the problem seems closely related to multi-agent TSP. Prior work has formulated the k-TSP solution in terms of splitting a single tour [15] or as an optimization task [16]. Some work [17] deals with asymmetric edge weights which are more relevant to the problems of our interest. The problem can be posed as an instance of multi vehicle pickup and delivery (PDP) [18]. Prior work [19] has applied the PDP problem to robots, taking into account time windows and robot-robot transfers. The seminal work [20], [21] has explored its complexity, and concedes to the hardness of the problem, while others studied cost bounds [22], and proposed ILP formulations [21]. Typically this line of work ignores coordination costs, though some methods [23] reason about it on candidate solutions.

Navigation among movable objects deals with the combinatorial challenges of multiple objects [24], [25], and has been shown to be a hard problem, and extended to manipulation applications [26]. Despite a lot of interesting work on challenges of manipulation and design choices [27] in grasp planning, the current work shall make assumptions that avoid complexities arising from them. The availability of manipulators opened the door to solve rearrangement tasks [28], [29], including instances where objects can be grasped only once or monotone [26], as well as non-monotone instances [30], [31]. Efficient solutions to assembly planning problems [32], [33] typically assume monotonicity, as without it the problem becomes much more difficult. Recent work has dealt with the hard instances of task planning [34], [35] and rearrangement planning [36], [37], [1]. Sampling-based task planning has made a recent push towards guarantees of optimality [38],

[39]. These are broader approaches that are invariant to the combinatorial structure of the application domain. The current work draws inspiration from these varied lines of research.

General task planning methods are unaware of the underlying structure studied in this work. Single-arm rearrangement solutions will also not be effective in this setting. The current work tries to bridge this gap and provide insights regarding the structure of two-arm rearrangement. Under assumptions that enable this study, an efficient solution emerges for this problem.

III. Problem Setup and Notation

Consider a planar surface and a set of n rigid objects $\mathcal{O} = \{o_1, o_2 \cdots o_n\}$ which can rest on the surface in stable poses $p_i \in \mathcal{P}_i \subset SE(3)$. The arrangement space $\mathcal{A} = \mathcal{P}_1 \times \mathcal{P}_2 \ldots \times \mathcal{P}_n$ is the Cartesian product of all $\mathcal{P}_i$, where every rearrangement $A \in \mathcal{A}$ is an n-tuple $(p_1, \ldots, p_n)$. In valid arrangements $\mathcal{A}_{\text{val}} \subset \mathcal{A}$, the objects are pairwise disjoint.

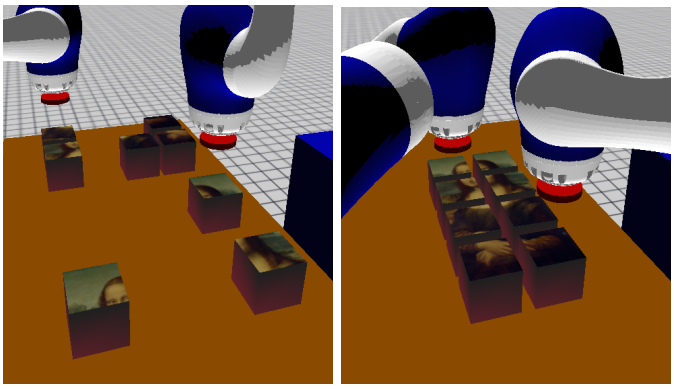


Fig. 2. An example of object rearrangement involving two robotic arms. Initial (left) and final (right) object configuration.

Two robot arms [1] m^1 and m^2 can pick and place the objects from and to the surface. $\mathbb{C}^k_{\text{free}}$ is the set of collision-free configurations for arm m^k (while ignoring possible collisions with objects or the other arm), and is assumed here to be a subset of d-dimensional Euclidean space $\mathbb{R}^d$. A path for m^k is denoted as $\pi^k : [0,1] \to \mathbb{C}^k_{\text{free}}$ and includes picking and placing actions throughout the entire rearrangement process. Let the set of collisions induced between the geometries of both the arms at two configurations along the path (at t) be denoted by the expression $\pi^1(t) \ominus \pi^2(t)$. If there are no collisions then $\pi^1(t) \ominus \pi^2(t) = \emptyset$. The space of dual-arm paths $\mathcal{D}$ is made of pairs of paths for the two arms: $D = (\pi^1, \pi^2) \in \mathcal{D}$. Then, $A(A_{\text{init}}, D)$ is the resulting arrangement when the objects are at A_{init} and moved according to D. Let $\text{cost}(D) : \mathcal{D} \to \mathbb{R}$ be a cost function over dual-arm paths.

Definition 1 (Optimal Two-arm Rearrangement): Given arms m^1, m^2 and objects $\mathcal{O}$ to be moved from initial arrangement $A_{\text{init}} \in \mathcal{A}_{\text{val}}$ to target arrangement $A_{\text{goal}} \in \mathcal{A}_{\text{val}}$, the optimal dual-arm rearrangement problem asks for a dual-arm path $D^* \in \mathcal{D}$, which optimizes a cost function, and solution paths belong to the valid configuration space.

$$D^* = (\pi^{*1}, \pi^{*2}) = \underset{\forall D \in \mathcal{D}}{\texttt{argmin}}\ \text{cost}(D) \qquad (1)$$
$$\text{such that } A(A_{\text{init}}, D^*) = A_{\text{goal}} \text{ ,and } \forall t \in [0,1]$$
$$\pi^{*1}(t) \in \mathbb{C}^1_{\text{free}},\ \pi^{*2}(t) \in \mathbb{C}^2_{\text{free}},\ \pi^{*1}(t) \ominus \pi^{*2}(t) = \emptyset$$

Assumptions are introduced to deal with the problem's combinatorics. The reachable task-space $\mathcal{T}^k \subset SE(3)$ of arm m^k is the set of $SE(3)$ poses that objects attached to the arm's end effector can acquire.

Let the ordered set of objects moved during the arm path π^k be denoted as $\bar{\mathcal{O}}(\pi^k)$. In general, an object can appear many times in $\bar{\mathcal{O}}(\pi^k)$. The current work, however, focuses on monotone instances, where each object is moved once.

Assumption 1 (Monotonicity): There are dual-arm paths $D = (\pi^1, \pi^2)$ that satisfy Eq. 1, where each object $o_i \in \mathcal{O}$ appears once in $\bar{\mathcal{O}}(\pi^1)$ or $\bar{\mathcal{O}}(\pi^2)$.

For the problem to be solvable, all objects are reachable by at least one arm at both A_{init} and A_{goal}: $\forall\ p_i \in A_{\text{init}}$ and $\forall\ p_j \in A_{\text{goal}}$ and $\exists\ k \in [1,2] : p_i, p_j \subset \mathcal{T}^k$. The focus will be on simultaneous execution of transfer and move paths.

Transfers: Dual-arm paths $T(\pi^1_i, \pi^2_i) \in \mathcal{D}$, where $\bar{\mathcal{O}}(\pi^k_i) = o^k_i$ and each m^k:

- starts the path in contact with an object o^k_i at its initial pose in A_{init},
- and completes it in contact with object o^k_i at its final pose in A_{goal}.

Moves or Transits: Paths $M(\pi^1_{i\to i'}, \pi^2_{i\to i'}) \in \mathcal{D}$, $\bar{\mathcal{O}}(\pi^k_{i\to i'}) = \emptyset$, and each m^k:

- starts in contact with object o^k_i at its final pose in A_{goal},
- and completes it in contact with object $o^k_{i'}$ at its initial pose in A_{init}.

Assumption 2 (Synchronicity): Consider dual-arm paths, which can be decomposed into a sequence of simultaneous transfers and moves for both arms:

$$D = (T(\pi^1_1, \pi^2_1), M(\pi^1_{1\to 2}, \pi^2_{1\to 2}), \ldots, M(\pi^1_{\ell-1\to\ell}, \pi^2_{\ell-1\to\ell}), T(\pi^1_\ell, \pi^2_\ell)). \qquad (2)$$

For simplicity, Eq. 2 does not include an initial move from $q^k_{\text{safe}} \in \mathbb{C}^k_{\text{free}}$ and a final move back to q^k_{safe} (an odd number of objects can also be easily handled). Then, the sequence of object pairs moved during a dual-arm path as in Eq. 2 is:

$$\Omega(D) = \Big(\omega_i = (o^1_i, o^2_i) \mid i, j \in [1 \cdots \ell], \bigcup_i (o^1_i \cup o^2_i) = \mathcal{O}, \forall k, k' \in [1,2], o^k_i \neq o^{k'}_j \Big). \qquad (3)$$

Given the pairs of objects ω_i, it is possible to express a transfer as $T(\omega_i)$ and a move as $M(\omega_{i\to j})$. Then, $D(\Omega)$ is the synchronous, monotone dual-arm path due to $\Omega = (\omega_1, \ldots, \omega_\ell)$, i.e., $D(\Omega) = (T(\omega_1), M(\omega_{1\to 2}), \ldots, M(\omega_{\ell-1\to\ell}), T(\omega_\ell))$.

Assumption 3 (Object Non-Interactivity): There are collision-free transfers $T(\omega_i)$ and moves $M(\omega_{i\to i'})$ regardless of the object poses in A_{init} and A_{goal}. This entails that there

[1] We shall use superscripts to denote arms, and subscripts to index into some sequence or data-structure dependent on context, unless stated otherwise.

is no interaction between the n resting objects and the arms during transits. Similarly, there are no interactions between the arm-object system and the $n-2$ resting objects during the transfers. Collisions involving the arms, static obstacles and picked objects are always considered.

The metric this work focuses on relates to makespan and minimizes the sum of the longest distances traveled by the arms in each synchronized operation. Let $\|\pi^k\|$ denote the Euclidean arc length in $\mathbb{C}^k_{\text{free}} \subset \mathbb{R}^d$ of path π^k. Then, for transfers $\text{cost}(T(\pi^1_i, \pi^2_i)) = \max\{\|\pi^1_i\|, \|\pi^2_i\|\}$. Similarly, $\text{cost}(M(\pi^1_{i\to i'}, \pi^2_{i\to i'})) = \max\{\|\pi^1_{i\to i'}\|, \|\pi^2_{i\to i'}\|\}$. Then, over the entire dual-arm path D define:

$$\text{cost}(D) = \sum_{i=1}^{\ell} \text{cost}(T(\omega_i)) + \sum_{i=1}^{\ell-1} \text{cost}(M(\omega_{i\to i+1})). \quad (4)$$

Note that the *transfer* costs do not depend on the order in which the objects are moved but only on the assignment of objects to arms. The *transit* costs arise out of the specific ordering in $\Omega(D)$. Then, for the setup of Definition 1 and under Assumptions 1-3, the problem is to compute the optimal sequence of object pairs Ω^* so that $D(\Omega^*)$ satisfies Eq. 1 and minimizes the cost of Eq. 4.

A. Discussion on Assumptions

This work restricts the study to a class of monotone problems that relate to a range of industrial packing and stowing applications. The monotonicity assumption is often used in manageable variants of well-studied problems [32], [33]. A monotone solution also implies that every object's start and target is reachable by at least one arm. Objects do not need to be in the commonly reachable workspace as long as the problem is monotone and solvable. Section VI discusses a `NO_ACT` task that can deal with unbalanced object assignments between two arms.

The synchronicity assumption allows to study the combinatorial challenges of the problem, which do not relate to the choice of time synchronization of different pickups and drop-offs. Section VI describes the use of `dRRT*`[10], [11] as the underlying motion planner that can discover solutions that can synchronize arm motions for simultaneous picks, and simultaneous placements. The synchronicity assumption is relaxed through smoothing (Section VI).

The non-interactivity assumption comes up naturally in planar tabletop scenarios with top-down picks or delta robots. Such scenarios are popular in industrial settings. Once the object is raised from the resting surface, transporting it to its target does not introduce interactions with the other resting objects. This assumption is also relaxed in Section VI, with a lazy variant of the proposed method. Once a complete candidate solution is obtained, collision checking is performed with the entire scene.

Overall, under Assumptions 1-3, the current work identifies a problem structure, which allows arguments pertaining to the search space, completeness, and optimality. Nevertheless, the smoothed, lazy variant of the proposed method will still, in the experiments that were carried out, often return effective solutions in practice, even if these assumptions do not hold.

Scope and Limitations: A variety of automation applications like packing, stowing, or sorting involve different singulated objects arriving at designated source regions, and require their rearrangement to a disjoint target region or container. Such automation applications also create semi-structured workspaces that guarantee that all objects of interest are easily graspable (top-down for planar surfaces). This class of problems is monotone since the source and target poses are disjoint, and *no transfer will require an intermediate location*. In this case object non-interactivity essentially implies that the workspace is well-behaved enough to allow feasible single-arm motion plans between top-down reachable poses of the object. The current work addresses what we need to do in order to go beyond single arm setups and in particular in order to effectively deploy two arms to speed up such tasks. Our work also theoretically motivates the use of two arms over a single arm in monotone object rearrangement.

The assumptions on the domain laid out in this work serve to identify the class of problems where our theoretical bounds will be close to the attained performance. *The lazy variant proposed in this work is assured to iterate over all possible orderings of object-to-arm assignments, and take into account planning feasibility of the same. As such it is guaranteed to discover a solution as long as the two-arm rearrangement problem is monotone.* Admittedly, in domains where these assumptions are pathologically violated the theoretical and practical gains highlighted in this work will be compromised. The proposed method has limiting applicability to some interesting domains where objects are occluded, overlap, or lie in unstructured piles. Any non-monotone problems, i.e., problems that necessitate using an intermediate location, cannot be addressed by the current pipeline. These include puzzle-like or highly cluttered scenarios.

IV. Baseline Approaches and Size of Search Space

This section highlights two optimal strategies to discover $D(\Omega^*)$: a) exhaustive search, which reveals the search space of all possible sequences of object pairs Ω and b) an MILP model. Both alternatives, however, suffer from scalability issues as for each possible assignment, it is necessary to solve a coordinated motion planning problem for the arms. This motivates minimizing the number of assignments ω considered, and the number of motion planning queries it requires for discovering $D(\Omega^*)$, while still aiming for high quality solutions.

A. Exhaustive Search

The exhaustive search approach, illustrated in Fig. 3 is a brute force expansion of all possible sequences of object pairs Ω. Nodes correspond to transfers $T(\omega_i)$ and edges are moves $M(\omega_{i\to i'})$. The approach evaluates the cost for all possible branches to return the best sequence Ω. The total number of possible distinct plans is $n!$.

$$ {}^nP_2 + {}^nP_2 \times {}^{n-2}P_2 + ... + {}^nP_2 \times {}^{n-2}P_2 \times {}^{n-4}P_2 ... \times {}^2P_2 = \sum_{L=1}^{n/2} \prod_{k=0}^{2L-1} (n-k) \quad (5)$$

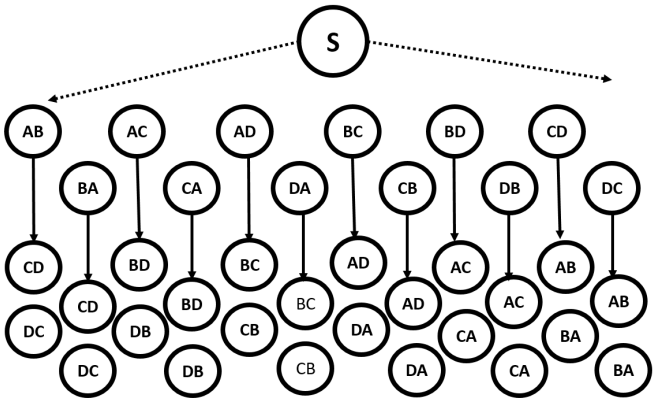


Fig. 3. Search tree for 4 objects (A,B,C,D) where each branch orders a sequence of object-arm assignments.

where ${}^nP_k$ is the k-permutations of n.

Motion plans can be reused, however, and repeated occurrences of $T(\omega_i)$ and $M(\omega_{i\to i'})$ should be counted only once for a total of:

– ${}^nP_2$ *transfers* of objects, and
– ${}^nP_2 \times {}^{n-2}P_2$ *transits* between all possible valid ordered pairs of ω.

Additional motion plans are needed for the initial move from q^k_{safe} and the return to it at the end of the process, introducing $2 \times {}^nP_2$ transits:

$$\begin{aligned}&\#\text{ of Transfers} + \#\text{ of Moves} = \\ &{}^nP_2 + \Big(({}^nP_2 \times {}^{n-2}P_2) + (2 \times {}^nP_2)\Big)\end{aligned} \tag{6}$$

This returns an optimal synchronized solution but performs an exhaustive search and requires exponentially many calls to a motion planner.

B. MILP Formulation

Mixed Integer Linear Programming (MILP) formulations can utilize highly optimized solvers [40]. Prior work has applied these techniques for solving m-TSP [16], [17] and pickup-and-delivery problems [19], [21], but viewed these problems in a decoupled manner. This work outlines an MILP formulation for the synchronized two-arm rearrangement problem that reasons about coordination costs arising from arm interactions in a shared workspace.

Graph Representation: The problem can be represented as a directed graph where:

- vertices $v = \omega_v = (o^1_v, o^2_v)$ correspond to a transfer $T(\omega_v)$;
- edges $e(u, v)$ are valid moves $M(\omega_{u\to v})$.

A valid edge $e(u, v)$ is one where an object does not appear more than once in the transfers of nodes u and v. The cost of a directed edge $e(u, v)$ encodes both the cost of the transfer $T(\omega_u)$ and the cost of the move $M(\omega_{u\to v})$. There is also a vertex S, which connects moves from and to the safe arm configurations q^k_{safe}. The directed graph $\hat{G}(\hat{V}, \hat{E})$ is defined:

$$\begin{aligned}\hat{V} =&\{S\} \cup \{v = \omega_v = (o^1_v, o^2_v) \mid \\ &\forall\ o^1_v, o^2_v \in \mathcal{O}, o^1_v \neq o^2_v\},\\ \hat{E} =&\{e(u, v) \mid \forall\ u, v \in \hat{V} \text{ so that } u \neq v,\\ &o^k_v \neq o^\ell_u\ \forall\ k, \ell \in [1, 2]\}\cup\\ &\{e(S, v)\ \forall\ v \in \hat{V} \setminus S\}\cup\\ &\{e(v, S)\ \forall\ v \in \hat{V} \setminus S\}\\ cost_{e(u,v)} =&\text{cost}(u) + \text{cost}(u, v)\\ =&\text{cost}(T(\omega_u)) + \text{cost}(M(\omega_{u\to v}))\end{aligned}$$

Let $\text{cost}(S) = 0$. The total number of motion planning queries needed to be answered to define the edge costs is given in Eq. 6. The formulation proposed in this section tries to ensure the discovery of Ω^* on $\hat{G}$ as a tour that starts and ends at S, while traversing each vertex corresponding to Ω^*. To provide the MILP formulation, define $\delta_{\text{in}}(v)$ as the in-edge set v, and $\delta_{\text{out}}(v)$ as the out-edge set. Then, $\gamma(o)$ is the object coverage set $\gamma(o) = \{e(u, v) \mid e \in \hat{E}, o \in \omega_u\}$, i.e., all the edges that transfer an object o.

Model: Set the optimization objective as:

$$\min \sum_{e\in\hat{E}} cost_e x_e \tag{A}$$

Eq. [B] below defines indicator variables. Eqs. [C-E] ensure edge-flow conserved tours. Eqs. [F-G] force S to be part of the tour. Eq. [H] transfers every object only once. Eq. [I] lazily enforces the tour to be of length $\frac{n}{2}+1$ [41]. While the number of motion-planning queries to be solved is the same as in exhaustive search, efficient MILP solvers [40] provide a more scalable search process.

$$x_e \in \{0, 1\}\quad \forall e \in \hat{E} \tag{B}$$

$$\sum_{e\in\delta_{\text{in}}(v)} x_e \leq 1\quad \forall v \in \hat{V} \tag{C}$$

$$\sum_{e\in\delta_{\text{out}}(v)} x_e \leq 1\quad \forall v \in \hat{V} \tag{D}$$

$$\sum_{e\in\delta_{\text{in}}(v)} x_e = \sum_{e\in\delta_{\text{out}}(v)} x_e\quad \forall v \in \hat{V} \tag{E}$$

$$\sum_{e\in\delta_{\text{in}}(S)} x_e = 1 \tag{F}$$

$$\sum_{e\in\delta_{\text{out}}(S)} x_e = 1 \tag{G}$$

$$\sum_{e\in\gamma(o)} x_e = 1\quad \forall o \in \mathcal{O} \tag{H}$$

$$\sum_{e(u,v)\in\mathfrak{T}} x_e < |\mathfrak{T}|\quad \forall \mathfrak{T} \subset \hat{V}, |\mathfrak{T}| \leq \frac{n}{2} \tag{I}$$

V. Efficient Solution via Tour over Matching

The optimal baseline methods described above highlight the problem's complexity. Both methods suffer from the large number of motion-planning queries they have to perform to compute the cost measures on the corresponding search

structures. For this purpose it needs to be seen whether it is possible to decompose the problem into solvable sub-problems.

A. Importance of Transfers

In order to draw some insight, consider again the tabletop setup with a general cost measure of c_t per unit distance. Observation 1 suggests that under certain conditions, there may not be a meaningful bound on the performance ratio between a k-arm solution and a single-arm solution. This motivates the examination of another often used setting—randomly chosen non-overlapping start and goal locations for n objects (within a unit square). In order to derive a meaningful bound on the benefit of using a 2-arm solution to a single-arm solution, we first derive a conservative cost of a single-arm solution. A single-arm optimal cost has three main parts: 1) the portion of the transfer cost involving the pickup and drop-off of the n objects with a cost of $C_{pd} = nc_{pd}$, 2) the remaining transfer cost from start to goal for all objects C_{sg}, and 3) the transit cost going from the goals to starts C_{gs}. The single arm cost is

$$C_{\text{single}} = nc_{pd} + C_{sg} + C_{gs}. \tag{7}$$

To estimate Eq. 7, first note that the randomized setup will allow us to obtain the expected C_{sg}[42] as $\frac{2+\sqrt{2}+5\ln(1+\sqrt{2})}{15}nc_t \approx 0.52nc_t$. Approximate C_{gs} by simply computing an optimal assignment of the goals to the starts of the objects excluding the matching of the same start and goal. Denoting the distance cost of this matching as C^M_{gs}, clearly $C_{gs} > C^M_{gs}$ because the paths produced by the matching may form multiple closed loops instead of the desired single loop that connects all starts and goals. However, the number of loops produced by the matching procedure is on the order of $\ln n$ and therefore, $C_{gs} < C^M_{gs} + c_t \ln n$, by [22]. By [43], $C^M_{gs} = \Theta(\sqrt{n \ln n})$. Putting these together, we have,

$$\Theta(c_t\sqrt{n \ln n}) = C^M_{gs} < C_{gs} < C^M_{gs} + c_t \ln n = \Theta(c_t\sqrt{n \ln n}) + c_t \ln n$$

which implies that $C_{gs} \approx C^M_{gs}$ because for large n, $\ln n \ll \sqrt{n \ln n}$. It is estimated in [44] that $C^M_{gs} \approx 0.44\sqrt{n \ln n}c_t$ for large n. Therefore,

$$\begin{aligned} C_{\text{single}} &\approx nc_{pd} + (0.52n + 0.44\sqrt{n \ln n})c_t \\ &\approx (c_{pd} + 0.52c_t)n \end{aligned} \tag{8}$$

noting that the $0.44\sqrt{n \ln n}c_t$ term may also be ignored for large n. The cost of dual arm solutions will be analyzed in Section VII. We summarize the discussion above in the following lemma.

Lemma 1: For large n, the transfers dominate the cost of the solution.

We formulate a strategy to reflect this importance of transfers. The proposed approach gives up on the optimality of the complete problem, instead focusing on a high-quality solution, which:

- first optimizes transfers and selects an assignment of object pairs to arms,
- and then considers move costs and optimizes the schedule of assignments.

This ends up scaling better by effectively reducing the size of the search space and performing fewer motion planning queries. It does so by optimizing a related cost objective and taking advantage of efficient polynomial-time algorithms.

B. Foundations

Consider a complete weighted directed graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$ (Eqs. 9), where $v \in \mathcal{V}$ corresponds to a single object o. Each directed edge, $e = (o_i, o_j) \in \mathcal{E}$ is an ordered pair of objects o_i and o_j, where the order determines the assignment of an object to an arm m^1 or m^2. The cost of an edge cost(e) is the coordinated motion planning cost of performing the transfer corresponding to $\omega = (o_i, o_j)$. For instance, as shown in Fig 4(top-middle), $e(A, B)$ corresponds to m^1 transferring 'A', while m^2 transfers 'B', and the cost$(e(A, B))$ = cost$(T(\omega = (A, B))$ is the cost of such a concurrent motion. It should also be noted that since the arms are different, in general, cost$(e(A, B)) \neq$ cost$(e(B, A))$.

$$\begin{aligned} &\mathcal{G}(\mathcal{V}, \mathcal{E}) \\ &\mathcal{V} = \{v = o, \forall o \in \mathcal{O}\} \\ &\mathcal{E} = \{e(u, v) : \omega = (u, v), \forall u, v \in \mathcal{V}\} \\ &\text{cost}(e(u, v)) = \text{cost}(T(\omega = (u, v))) \end{aligned} \tag{9}$$

By Eq. 4, the cost of the transfers can be reasoned about independent of their order. Define $\{\Omega\}$ as the unordered set of $\omega \in \Omega$, then unordered transfer cost component corresponds to $\sum_{\omega \in \{\Omega\}} \text{cost}(T(\omega))$. Since $\mathcal{G}$ is a complete graph (shown in Fig 4(top-right)) where every edge corresponds to every possible valid ω, by construction $\{\Omega\} \subset \mathcal{E}$. A candidate solution of a monotone two-arm rearrangement problem must transfer every object exactly once. In terms of the graph, this means that in the subset of edges $\{\Omega\}$ every vertex appears in only one edge. We arrive at the following crucial observation.

Lemma 2 (Perfect Matching): Every candidate solution to a monotone dual-arm rearrangement problem comprises a set of unordered object-to-arm assignments $\{\Omega\}$ that is also a perfect matching solution on $\mathcal{G}$.

As per the decomposition of the costs in Eq. 4, it follows that the cost$(\{\Omega\})$ is the cost of the transfers in the solution. The solution to the minimum-weight perfect edge matching problem on such a graph would correspond to a $\{\Omega\}$ that optimizes the cost of *transfers* of all the objects.

Lemma 3 (Optimal Matching): The set of object-to-arm assignments $\{\Omega\}^*$ that minimizes the cost of object transfers is a solution to the minimum-weight perfect matching problem on $\mathcal{G}$.

Once such an optimal assignment $\{\Omega\}^*$ is obtained, the missing part of the complete solution is the set of transits between the object transfers and their ordering. Construct another directed graph $\mathcal{G}_\Gamma$ (Eqs. 10), where the vertices comprise of the $\omega \in \{\Omega\}^*$, i.e., the optimal object-to-arm assignments. An edge $e(\omega_{u \to v})$ corresponds to the coordinated *transit* motion between them. For instance, as in Fig 4(bottom-middle) for

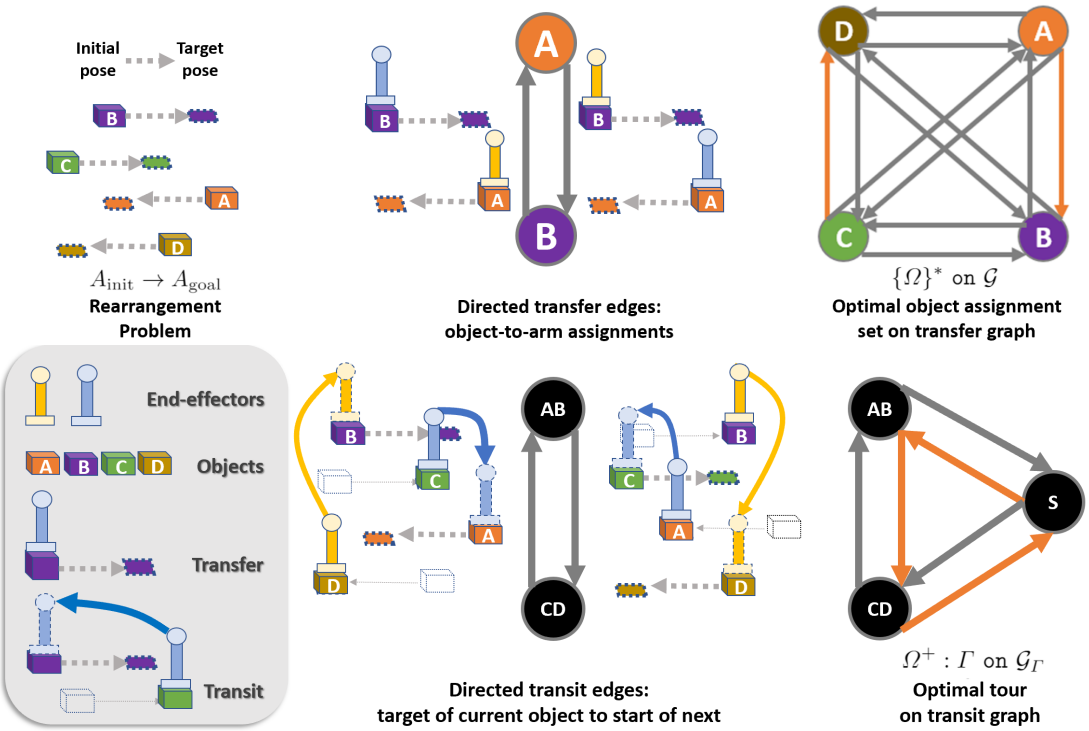


Fig. 4. *(Top-left)* A two-arm rearrangement problem. *(Top-middle)* Directed edges between objects represents the assignment of arms to object *transfers*. *(Top-right)* The problem of optimizing object transfers reduces to a minimum weight edge matching on a fully connected directed graph of *transfers*. *(Bottom-left)* The legends for the different symbols used in the diagrams. *(Bottom-middle)* A directed edge between object pairs represents the *transit* between the target poses of the previous objects to the initial poses of the next. *(Bottom-right)* Each transit moves the arms from the target poses of the current objects, to the initial poses of the next objects in the object-arm assignments. A sequence of object-arm assignments form an optimal tour over a *transit* graph.

an edge between $\omega(A, B)$ and $\omega(C, D)$, m^1 moves from the target pose of object 'A' to the starting pose of object 'C', and similarly m^2 moves from the target of 'B' to the start of 'D'. An additional vertex corresponding to the starting (and ending) configuration of the two arms (S) is added to the graph. The graph (shown in Fig 4(bottom-right)) is fully connected again to represent all possible transits or moves.

$$\begin{aligned} \mathcal{G}_\Gamma(\mathcal{V}_\Gamma, \mathcal{E}_\Gamma) \\ \mathcal{V}_\Gamma &= \{v = \omega, \forall \omega \in \{\Omega\}^*\} \cup \{\omega_S\} \\ \mathcal{E}_\Gamma &= \{e(u, v) : \omega_{u \rightarrow v}, \forall u, v \in \mathcal{V}_\Gamma\} \\ \text{cost}(e(u, v)) &= \text{cost}(M(\omega_{u \rightarrow v})) \end{aligned} \tag{10}$$

A complete candidate solution to the problem now requires the sequence of ω, which is a complete tour over $\mathcal{G}_\Gamma$, that visits all the vertices i.e., an ordered sequence of vertices $\Gamma = (S, \Omega_\Gamma, S)$.

Lemma 4 (Tour): Any complete tour Γ over the graph $\mathcal{G}_\Gamma$, corresponds to a sequence of object-to-arm assignments Ω_Γ and is a candidate solution to the synchronous two-arm rearrangement problem.

Let $\mathbb{P}^{\{\Omega\}}$ represent the set of all possible ordering of the elements in $\{\Omega\}$. This means, any candidate tour corresponds to a $\Omega_\Gamma \in \mathbb{P}^{\{\Omega\}}$. An optimal tour on $\mathcal{G}_\Gamma$ minimizes the transit costs over the all possible candidate solutions in $\mathbb{P}^{\{\Omega\}^*}$

$$\Omega^+ = \underset{\Omega \in \mathbb{P}^{\{\Omega\}^*}}{\texttt{argmin}} \sum_{e(u,v) \in \Gamma} \text{cost}(M(\omega_{u \rightarrow v}))). \tag{11}$$

This differs from the true optimal Ω^*, since the second step of finding the optimal transit tour only operates over all possible solutions that include the optimally matched transfers obtained in the first step. The insight here is that, even though Ω^+ reports a solution to a hierarchical optimization objective, the search space is much smaller, and the sub-problems more efficient to solve.

Algorithm 1: `TOM`$(\mathcal{O}, S, A_{\text{init}}, A_{\text{goal}})$

```
G ← transfer_graph(O, A_init, A_goal);
{Ω}* ← optimal_matching(G);
G_Γ ← transit_graph({Ω}*, S);
Ω+ ← optimal_tour(G_Γ);
return D(Ω+);
```

C. Algorithm : Tour-Over-Matching

This section describes the algorithm (`TOM`) outlined in the previous section. The steps correspond to Algo 1.

- `transfer_graph`: This function constructs a directed graph $\mathcal{G}$ defined by Eqs. 9. This step creates a graph with n vertices and ${}^nP_2$ edges.
- `optimal_matching`: This function takes the graph $\mathcal{G}$ constructed as an argument and returns the unordered set of edges, corresponding to the set of optimal transfers over $\mathcal{G}$. Optimal matching over an undirected graph can be solved using Edmond's Blossom Algorithm [14], [45], [46]. The directed graph $\mathcal{G}$ is converted into an equivalent undirected graph $\mathcal{G}_U$. Since $\mathcal{G}$ is complete,

every pair of vertices shares two directed edges. $\mathcal{G}_U$ only preserves the minimally weighted connection for every vertex pair. The result of matching is a subset of edges on $\mathcal{G}_U$ which correspond to a set of directed edges on $\mathcal{G}$ i.e., $\{\Omega\}^*$. The runtime complexity of the step is $O(|\mathcal{E}||\mathcal{V}|\log|\mathcal{V}|) = O({}^nP_2 n\log n) = O(n^3\log n)$.

- `transit_graph`: This function constructs the directed transit graph $\mathcal{G}_\Gamma$ as per the set of Eqs. 10. This constructs $\frac{n}{2}+1$ vertices and ${}^{(\frac{n}{2}+1)}P_2$ edges.
- `optimal_tour`: Standard TSP solvers like Gurobi [40] can be used to find the optimal tour over $\mathcal{G}_\Gamma$ corresponding to Ω^+.

The costs are deduced from coordinated motion plans over edges. The total number of such calls compared to the count from the baseline in Equ. 6, shows a saving of a factor of $O(n^2)$ queries (# of Transfers + # of Moves).

$$\frac{\text{Baseline } \#}{\texttt{TOM } \#} = \frac{{}^nP_2 + \left({}^nP_2 \times {}^{n-2}P_2\right) + \left(2 \times {}^nP_2\right)}{{}^nP_2 + {}^{(\frac{n}{2}+1)}P_2} = \frac{4(n-1)((n-5)n+9)}{5n-2} \tag{12}$$

The evaluation performed here focuses on the maximum of distances (Eq. 4) covered by the arms in a solution for a fair comparison with the other methods. The prioritization of optimization objective, however, is also amenable to other cost functions, where carrying objects is often more expensive than object-free motions.

VI. Integration with Motion Planning

The algorithms described so far are agnostic to the underlying motion planner. Depending upon the model of the application domain, different motion planning primitives might be appropriate. For planar environments with disk robot pickers (similar to delta robots), recent work [47] characterizes the optimal two-disk coordinated motions. The current implementation uses a general multi-robot motion planning framework `dRRT*` [10], [11] for dual-arm coordinated planning.

In practice the cost of generating and evaluating two-arm motions can dominate the overall running time of the algorithm, when compared to the combinatorial ingredients that discover the high-level plan, i.e., execution order and and arm assignment. Even though `TOM` reduces this, further improvements can be made with lazy evaluation.

Lazy Evaluation: Recent work [48] introduces heuristics for `dRRT*`, which pre-process estimates of the shortest path costs for the arms. Two-arm rearrangement can be significantly sped up if the motion planning queries are replaced with look-ups of such heuristics. Once a candidate Ω is obtained, motion planning can expand the solution $D(\Omega)$ to obtain the actual solution paths. If this fails, the algorithm tries other sequences.

The algorithm Algo 2 takes as input the algorithm `ALGO`, a heuristic $\mathcal{H}$, and a motion planner `MP`. $\mathcal{E}_\mathtt{b}$ keeps track of the blocked edges. The process keeps generating candidate solutions using the `ALGO` (Line 3). Line 5 motion plans over the candidate solution, and appends to the result (Line 6). Any failures are recorded in $\mathcal{E}_\mathtt{b}$ (Lines 8,10), and inform subsequent runs of `ALGO`.

Algorithm 2: `Lazy_Evaluation`(`ALGO`, $\mathcal{H}$, `MP`)

```
1  E_b ← ∅; D ← ∅;
2  while D = ∅ ∧ time_not_exceeded do
3  |   Ω ← ALGO(H, E_b);
4  |   for ω_i, ω_{i→i+1} ∈ Ω do
5  |   |   D_i, D_{i→i+1} ← MP(ω_i), MP(ω_{i→i+1});
6  |   |   D ← (D, D_i, D_{i→i+1});
7  |   |   if D_i = ∅ then
8  |   |   |   E_b ← E_b ∪ {ω_i}; D ← ∅;
9  |   |   if D_{i→i+1} = ∅ then
10 |   |   |   E_b ← E_b ∪ {ω_{i→i+1}}; D ← ∅;
11 |   |   if D = ∅ then break ;
12 return D
```

Completeness: The lazy variants give up on optimality for the sake of efficiency but given enough retries the algorithms will eventually solve every problem that `ALGO` can. The motion planning and evaluation is performed in sequence of the order of execution. The object non-interactivity assumption is relaxed since all the object positions are effectively known during the evaluation of a candidate solution.

Smoothing: Applying velocity tuning over the solution trajectories for the individual arms relaxes the synchronization assumption by minimizing any waits that might be a by-product of the synchronization. Smoothing does not change the maximum of distances, only improves execution time. Indications that the smoothed variants of the synchronous solutions do not provide significant savings in execution time are included in the results for the interested reader in Fig 9.

It should be noted that in an iterative version of `TOM`, in order to explore variations in optimal object-to-arm assignments $\{\Omega\}^*$ if failures occur in finding Γ, some edges need to be temporarily blocked on $\{\Omega\}^*$. The search structures can also be augmented with `NO_ACT` tasks, for possible ω where one of the arms do not move.

VII. Bounding Costs under Planar Disc Manipulator Model

This section studies the dual arm costs in the randomized unit tabletop setting, where c_t is the cost measure per unit distance. Assume for simplicity that each arm's volume is represented as a disc of some radius r. Eq. 8 derives the cost estimate for a single arm solution as approximately $(c_{pd} + 0.52c_t)n$. Firstly the following arguments can be made for k-arms.

Theorem 1: A k-arm solution can have an asymptotic improvement of $\frac{1}{k}$ over the single arm solution, when rearranging objects with non-overlapping starts and goals that are uniformly distributed in a unit square.

Proof: In the planar unit-square setting, with k arms, there are $\frac{n}{k}$ objects for each arm to work with. Consider the transfers and transits of a set of k objects, one for each arm. By [49], the arbitrary rearrangement of k discs can be achieved in a bounded region with a perimeter of $O(kr)$. Clearly, the per robot additional (makespan or distance) cost is bounded by some function $f(k,r)c_t$, which goes to zero as r goes to zero.

Adding up all the potential cost that can be incurred, a k-arm solution has a cumulative cost of:
$C_{\text{k-arm}} = C_{\text{single}} + nf(k,r)c_t \approx (c_{pd} + 0.52c_t + f(k,r)c_t)n$.
For fixed k and small r, $C_{\text{k-arm}}$ is almost the same as C_{single}. Upon considering the maximum of the two arc lengths or makespan, the k-arm cost becomes $C^t_{\text{k-arm}} \approx (c_{pd} + 0.52c_t)\frac{n}{k} + nf(k,r)c_t$.

The cost ratio is

$$\frac{C^t_{\text{k-arm}}}{C_{\text{single}}} \approx \frac{(c_{pd} + 0.52c_t)\frac{n}{k} + nf(k,r)c_t}{(c_{pd} + 0.52c_t)n} = \frac{1}{k} + \frac{f(k,r)c_t}{c_{pd} + 0.52c_t} \ . \tag{13}$$

When r is small or when $\frac{c_t}{c_{pd}}$ is small, the k-arm solution is roughly $\frac{1}{k}$ as costly as the single arm solution. On the other hand, in this model a k-arm solution does not do better than $\frac{1}{k}$ of the single arm. □

For obtaining a 2-arm solution, first partition the n objects randomly into two sets of $\frac{n}{2}$ objects each. Then, obtain the two initial solutions similar to the single arm case.

Corollary 1: A 2-arm solution can have an asymptotic improvement of $\frac{1}{2}$ over the single arm solution for rearranging objects with non-overlapping starts and goals that are uniformly distributed in a unit square

Proof: From the initial 2-arm solution, we construct an *asynchronous* 2-arm solution that is collision-free. Assume that pickups and drop-offs can be achieved without collisions between the two arms, which can be achieved with properly designed end-effectors. The main overhead is then the potential collision between the two (disc) arms during transfer and move operations. Because there are $\frac{n}{2}$ objects for each arm to work with, an arm may travel a path formed by $n+1$ straight line segments. Since we can have at most four intersections for the transfers and transits associated with a pair of objects (one for each arm), there are at most $2n$ potential collisions to handle. For each intersection, let one arm wait while the other circles around the first arm, which incurs a cost that is bounded by $2\pi \cdot r \cdot c_t$.

Adding up all the potential cost that a 2-arm solution can incur, the following cumulative cost is obtained:
$C_{\text{dual}} = C_{\text{single}} + 2n(2\pi r c_t) \approx (c_{pd} + 0.52c_t + 4\pi r c_t)n$.
For small r, C_{dual} is almost the same as C_{single}, and c_t is a distance (e.g., energy) cost. Upon considering the maximum of the two arc lengths or makespan (Eq. 4), the 2-arm cost becomes $C^t_{\text{dual}} \approx (c_{pd} + 0.52c_t)\frac{n}{2} + 4n\pi r c_t$.

The cost ratio is

$$\frac{C^t_{\text{dual}}}{C_{\text{single}}} \approx \frac{(c_{pd} + 0.52c_t)\frac{n}{2} + 4n\pi r c_t}{(c_{pd} + 0.52c_t)n} = \frac{1}{2} + \frac{4\pi r c_t}{c_{pd} + 0.52c_t} \tag{14}$$

When r is small or when $\frac{c_t}{c_{pd}}$ is small, the 2-arm solution is roughly half as costly as the single arm solution. On the other hand, in this model a 2-arm solution does not do better than $\frac{1}{2}$ of the single arm solution. □

Theorem 2: For rearranging objects with non-overlapping starts and goals that are uniformly distributed in a unit square, a randomized 2-arm synchronized solution can have an asymptotic improvement of $\frac{1}{2}$ over the single arm solution if $\frac{c_t}{c_{pd}}$ is small, and an improvement of roughly 0.64 when both c_{pd} and r are small.

Proof: In preparation for the proof of Theorem 2, we develop the following lemma.

Lemma 5: The expected measure of the maximum of lengths of two random lines on a unit square is 0.66.

Proof: Prior work [50] defines the *probability distribution function (pdf)* of lengths(ℓ) of randomly sampled lines in a rectangle.

Substituting the values for the dimensions of the rectangle in the unit square model, the *pdf* can be simplified as follows.

$$p(\ell) = 2\pi\ell - 8\pi\ell^2 + 2\ell^3, \quad \ell \in [0,1]$$
$$p(\ell) = 4\ell \sin^{-1}\left(\frac{1}{\ell}\right) - 4\ell\cos^{-1}\left(\frac{1}{\ell}\right) + 8\ell\sqrt{\ell^2 - 1} - 2\ell^3 - 4\ell, \quad \ell \in [1,\sqrt{2}],$$

where ℓ is the length measure, and $p(\ell)$ is the probability measure over different lengths. Assuming two random sets of lines, representing transfers in a random split of objects between two arms, the expected value of the maximum of these pairwise lengths i.e., $\mathtt{E}(\max(\ell_1, \ell_2)), \ \ell_1, \ell_2 \ i.i.d, \ \ell_1 \sim p, \ell_2 \sim p$, can be estimated using the *pdf* obtained.

$$\mathtt{E}(\max(\ell_1,\ell_2)) = \int_0^{\sqrt{2}}\int_0^{\sqrt{2}} \max(\ell_1,\ell_2)p(\ell_1)p(\ell_2)d\ell_2 d\ell_1 \approx 0.663$$

Prior work [42] offered an estimate for the expected length of a transit path C_{sg} in terms of the expected length of a line segment, 0.52, in a randomized setting in a unit square. With the current estimate of 0.66 for the maximum of two such randomly sampled line segments, it follows that, the expected makespan or maximum of distances cost will use this estimate. □

Now all the tools necessary for the proof of Theorem 2 are available.

The synchronization assumption changes the expected cost of the solution. The random partitioning of the n objects into two sets of $\frac{n}{2}$ object with a random ordering of the objects yields $\frac{n}{2}$ pairs of objects transfers, which dominate the total cost for large n. The cost (Eq. 4) of $\frac{n}{2}$ synchronized transfers (ω_i) includes $\frac{n}{2}c_{pd}$ and $C^{\text{sync}}_{sg} \approx (\mathtt{E}(\max(\ell_1,\ell_2))c_t)\frac{n}{2}$, where $\mathtt{E}(\max(\ell_1,\ell_2))$ is the expected measure of the maximum of lengths ℓ_1,ℓ_2 of two randomly paired transfers.

It follows that $C^{\text{sync}}_{\text{dual}} \approx (c_{pd} + 0.66c_t)\frac{n}{2} + 4n\pi r c_t$. The synchronized cost ratio is

$$\frac{C^{\text{sync}}_{\text{dual}}}{C_{\text{single}}} \approx \frac{(c_{pd} + 0.66c_t)\frac{n}{2} + 4n\pi r c_t}{(c_{pd} + 0.52c_t)n} = \frac{1}{2} + \frac{(0.07 + 4\pi r)c_t}{c_{pd} + 0.52c_t} \ . \tag{15}$$

When $\frac{c_t}{c_{pd}}$ is small, even the synchronized 2-arm solution provides an improvement of $\frac{1}{2}$. For the case when both r and c_{pd} are small, we observe that the ratio approaches 0.636. □

As a way to validate our asymptotic estimate, randomized trials were run with different number of randomly sampled

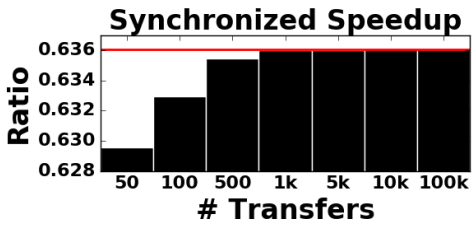


Fig. 5. Empirical cost ratio $\frac{C^{sync}_{dual}}{C_{single}}$ when $c_{pd} = 0, r = 0$ in the unit square model, versus the estimate (red line). As the number of object transfers increases the measured value converges to the estimate.

object transfer coordinates on an unit square. When $c_{pd} = 0$ and $r = 0$, the ratio of $\frac{C^{sync}_{dual}}{C_{single}}$ evaluates to 0.636. Fig 5 verifies empirically that the ratio converges to the expected value as the number of transfers increases. This indicates the asymptotic speedup of a synchronized dual arm solution for a makespan or maximum of distances cost metric.

Note on bounds: Even though the proposed simplified model may not be immediately suitable for general configuration spaces, experiments indicate that the speedups exist in these spaces as well.

VIII. Evaluation

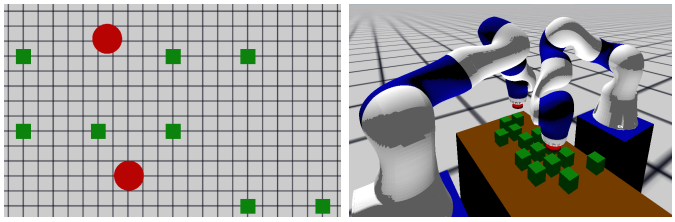


Fig. 6. *Picker* and *Manipulator* trials.

This section describes the experiments performed to evaluate the algorithms in two domains shown in Fig 6: a) simple picker and b) general manipulators. In order to ensure monotonicity, the object starts and goals do not overlap. Uniform cuboidal objects simplify the grasping problem, though this is not a limitation of the methods. 50 random experiments were limited to $300s$ of computation time. The underlying dRRT* motion planner is restricted to a max of $3s$ per plan. A comparison point includes a random split method, which splits $\mathcal{O}$ at random into two subsets and chooses an arbitrary ordering. Maximum of distances cost is compared to the single arm solution [1]. Computation times and success rates are reported. The trends in both experiments show that in the single-shot versions, exhaustive and MILP tend to time-out for larger n. Lazy variants scale much better for all the algorithms, and in some cases increase the success ratio due to retries. TOM has much better running time than exhaustive and MILP, and produces better and more solutions than random split. Overall, the results show that a) our MILP succeeds more within the time limit than exhaustive, b) TOM scales the best among all the methods, and c) **the cost of solutions from TOM is close to the optimal baseline**, which is around half of the single arm cost.

Note that we choose to focus on balanced problem instances where at least half of the objects are reachable to one of the robots. Unbalanced instances can be dealt with using NO_ACT object-to-arm assignments that require no dual-arm coordination, while the proposed method remains unchanged.

A. Simple Picker

This benchmark evaluates two disk robots hovering over a planar surface scattered with objects. The robots are only free to move around in a plane parallel to the resting plane of the objects, and the robots can pick up objects when they are directly above them. This benchmark is reminiscent of delta-robots operating over a picking surface. Fig 7(*top*) all runs up to 24 objects succeeded for TOM. MILP scales better than exhaustive. Lazy random split succeeds in all cases (*bottom*). In terms of solution costs (*middle*) exhaustive finds the true optimal. MILP matches exhaustive and TOM is competitive. In all experiments, TOM enjoys a success rate of 100% while having much better computation time that exhaustive and MILP, as the number of objects increases.

B. General Dexterous Manipulator

The second benchmark sets up two Kuka arms across a table with objects on it. The objects are placed in the common reachable part of both arms' workspace, and only one top-down grasping configuration is allowed for each object pose. For the Kuka arms top-down reachability exists in an annular region between 40-70cm from the base of each robot. The robots were placed 1.1m apart. Experiments were performed on the largest rectangle that fitted within the intersection of the annular reachability regions.

Here (Fig 8) a larger number of motion plans tend to fail, so the single shot variants show artifacts of the randomness of dRRT*in their success rates. Random split performs the worst since it is unlikely to chance upon valid motion plans. Single shot exhaustive and MILP scale poorly because of expensive motion planning. Interestingly, motion planning infeasibility reduces the size of the exhaustive search tree. The solution costs *(middle)* substantiate benefits of the use of two arms. The computation times *(bottom)* again show the scalability of TOM, even compared to random split.

C. Smoothing

The result of the velocity tuning over the solution trajectories for the individual arms as a post-processing step is shown in Fig 9. The objective is to minimize any waits that might be a by-product of the synchronization. The small % improvements indicate that the asynchronous variants of the solutions discovered from the methods do not yield a big enough saving in execution time. Most of the improvement as a percentage of the original solution duration is not too high. On top of that, the time taken to smooth the solutions for TOM (overlaid on Fig 9) shows that the trade-off is sometimes not beneficial. In their largest problem instances the Kuka spent $0.44s$ of smoothing time to save $3.23s$ off the solution duration, while the picker spent $9.84s$ to save $0.54s$.

This indicates that among the class of synchronized solutions discovered by the proposed algorithms, the asynchronous, smoothed variants do not seem to be drastically

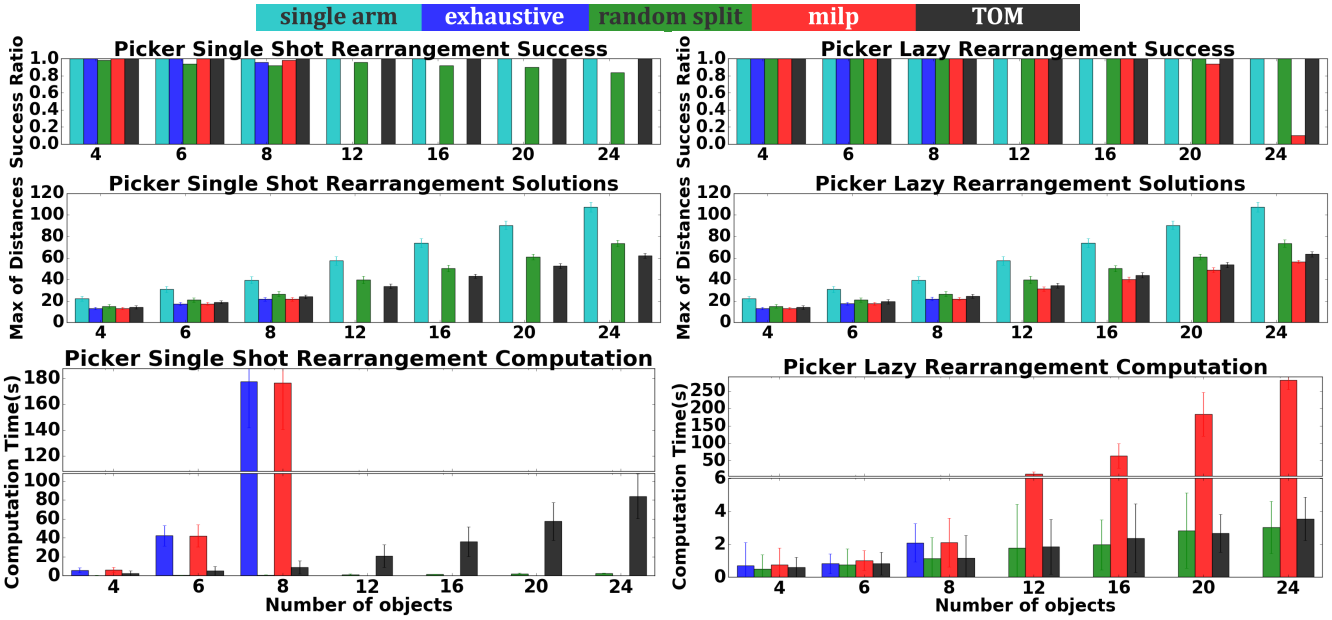


Fig. 7. *Simple Picker* results with success*(top)*, solution costs*(middle)*, and computation*(bottom)* reported for single-shot*(left)* and lazy*(right)* versions of the methods

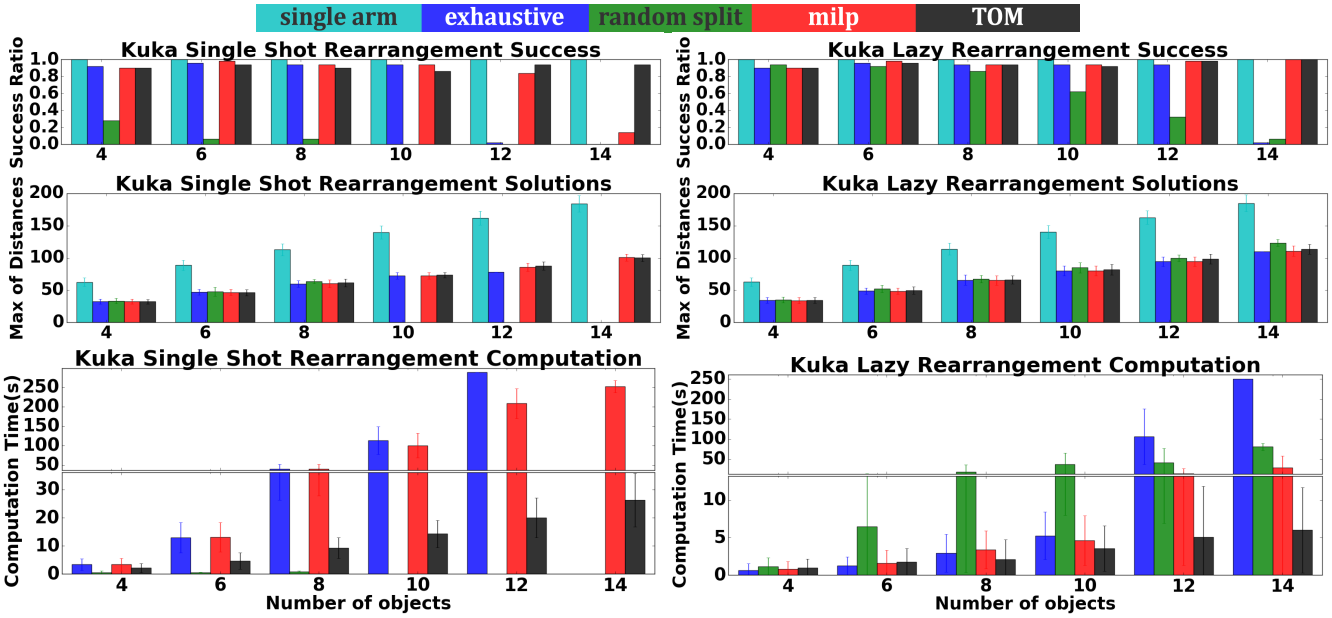


Fig. 8. `Kuka` results with success*(top)*, solution costs*(middle)*, and computation*(bottom)* reported for single-shot*(left)* and lazy*(right)* versions of the methods

better. Moreover, smoothing does not improve the maximum of distances cost measure, but only reduces the solution duration.

D. Reachability Benchmark

In this benchmark, two Kuka arms are placed opposite a target arrangement table, as shown in Fig 10. The initial poses of the objects on either side of the arms in a way that the initial poses are reachable by only one of the arms. The purpose of this study is to see the effects of general divided workspaces where both the initial and target poses are not in the region of common reachability of the arms. It should be noted that a single arm solution does not exist for this benchmark.

The problem of optimizing the assignment of arms to objects will be affected by the reachability. Expectedly, the naive random split tends to erroneously assign unreachable objects to arms. The data reported for the lazy variants, shows that our proposed method manages to maintain scalability and robustness even in such scenarios and moreover, further highlights the benefits of two-arm rearrangement.

Note on Heuristic Strategies: It is possible that instead of the random baseline, some other heuristic might be used to allocate the objects to arms, like the proximity of objects to arms or some workspace partitioning. Such a deterministic, decoupled heuristic will make it faster to compute object-arm assignments than the proposed reasoning, but trade off the optimality bounds posited in this work. Given the object-to-arm assignments feasibility of the solution ultimately depends on coordinated motion planning. It should be noted that committing to a single choice of object assignments seriously affects success rates (especially in the Kuka benchmark). The

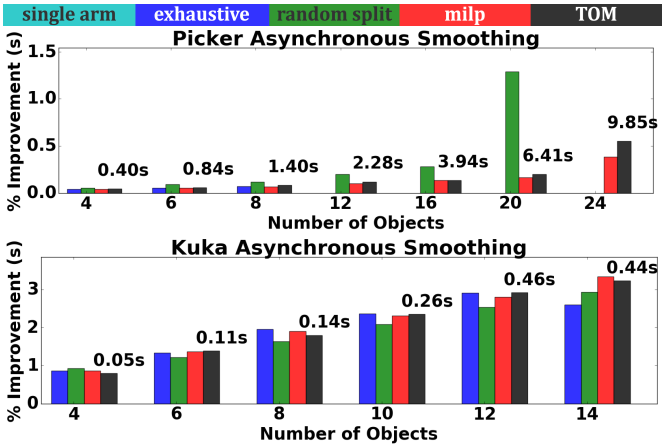


Fig. 9. Smoothed solution improvement as a percentage of the original synchronized solution duration, and the time taken to smooth solutions obtained from TOM in seconds.

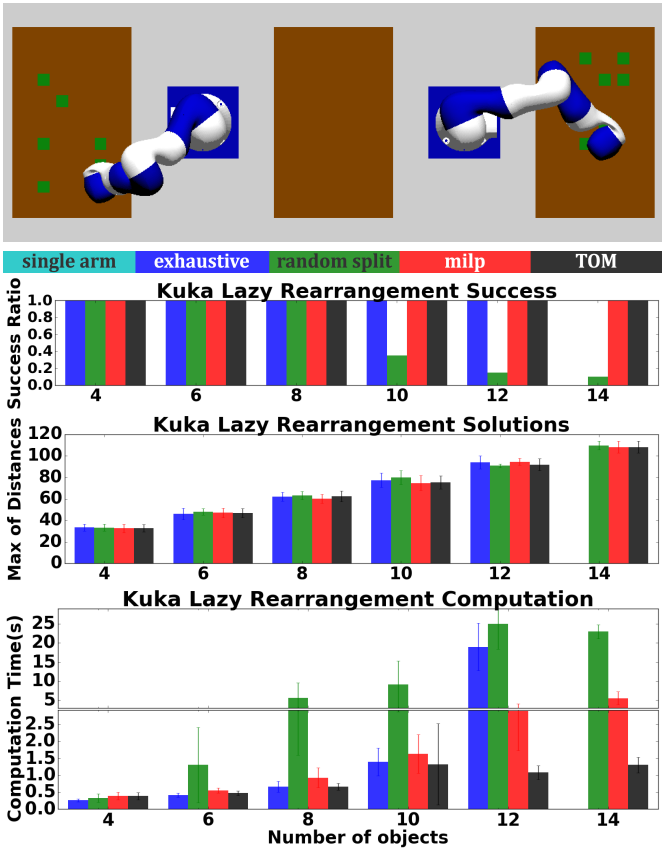


Fig. 10. The reachability benchmark with the initial poses of the objects lying in disjoint parts of the workspace(tables on the right and left) that are only reachable by one of the arms. The objective is to transfer the objects to the middle table.

success is only increased using our proposed lazy approach, which takes into account such infeasibility and keeps reattempting other alternatives. A deterministic heuristic however will not offer alternative object-arm assignments and thus is incompatible with the improvements obtained from lazy evaluation. As such we leave the learning and incorporation of appropriate heuristics in this domain as future work and a motivation for practitioners.

IX. Discussion

The current work demonstrates the underlying structure of synchronized two-arm rearrangement and proposes an MILP formulation, as well as a scalable algorithm TOM that provides fast, high quality solutions. Existing efficient solvers for reductions of the dual-arm problem made TOM effective.

Work beyond the methods presented in this paper has explored the k-arm case. The matching sub-problem ceases to have effective solvers for $k > 2$, and heuristics might need to be considered for feasible solutions. Effective approximate or lazy solutions to the k-arm case can also prove to be power heuristics to the general multi-arm task planning [51] challenge. Regrasp reasoning and the use of intermediate locations can extend this work to non-monotone problem instances. The incorporation of manipulation and grasp reasoning with real-world object geometries can also extend such solvers to more cluttered environments. This work serves as a stepping stone in building towards these rich problem domains.

References

[1] S. D. Han, N. Stiffler, A. Krontiris, K. Bekris, and J. Yu, "Complexity results and fast methods for optimal tabletop rearrangement with overhand grasps," *IJRR*, accepted 2018.

[2] R. Shome, K. Solovey, J. Yu, K. E. Bekris, and D. Halperin, "Fast and high-quality dual-arm rearrangement in synchronous, monotone tabletop setups," in *Workshop on the Algorithmic Foundations of Robotics (WAFR)*, Mérida, México, 12/2018 2018. [Online]. Available: http://www.cs.rutgers.edu/~kb572/pubs/Fast_High_Quality_Dual_Arm_Rearrangement.pdf

[3] K. Solovey, J. Yu, O. Zamir, and D. Halperin, "Motion planning for unlabeled discs with optimality guarantees," in *Robotics: Science and Systems XI, Sapienza University of Rome, Rome, Italy, July 13-17, 2015*, 2015. [Online]. Available: http://www.roboticsproceedings.org/rss11/p11.html

[4] K. Solovey and D. Halperin, "On the hardness of unlabeled multi-robot motion planning," *IJRR*, vol. 35, no. 14, 2016.

[5] J. P. Van Den Berg and M. H. Overmars, "Prioritized motion planning for multiple robots," in *IROS*, 2005.

[6] S. Leroy, J.-P. Laumond, and T. Siméon, "Multiple path coordination for mobile robots: A geometric algorithm," in *IJCAI*, vol. 99, 1999.

[7] G. Wagner, M. Kang, and H. Choset, "Probabilistic path planning for multiple robots with subdimensional expansion," in *ICRA*, 2012.

[8] M. Gharbi, J. Cortés, and T. Siméon, "Roadmap Composition for Multi-Arm Systems Path Planning," in *IROS*, 2009.

[9] K. Solovey, O. Salzman, and D. Halperin, "Finding a needle in an exponential haystack: Discrete RRT for exploration of implicit roadmaps in multi-robot motion planning," *IJRR*, vol. 35, no. 5, 2016.

[10] A. Dobson, K. Solovey, R. Shome, D. Halperin, and K. E. Bekris, "Scalable Asymptotically-Optimal Multi-Robot Motion Planning," in *MRS*, 2017.

[11] R. Shome, K. Solovey, A. Dobson, D. Halperin, and K. E. Bekris, "dRRT*: Scalable and informed asymptotically-optimal multi-robot motion planning," *Autonomous Robots*, Jan 2019. [Online]. Available: https://www.cs.rutgers.edu/~kb572/pubs/drrt_star_auro.pdf

[12] A. Adler, M. de Berg, D. Halperin, and K. Solovey, "Efficient multi-robot motion planning for unlabeled discs in simple polygons," *IEEE T-ASE*, vol. 12, no. 4, 2015.

[13] J. Yu and S. M. LaValle, "Optimal multirobot path planning on graphs: Complete algorithms and effective heuristics," *IEEE Transactions on Robotics*, vol. 32, no. 5, 2016.

[14] J. Edmonds, "Maximum matching and a polyhedron with 0, 1-vertices," *Journal of research of the National Bureau of Standards B*, vol. 69, no. 125-130, 1965.

[15] G. N. Frederickson, M. S. Hecht, and C. E. Kim, "Approximation algorithms for some routing problems," in *FOCS*, 1976.

[16] S. Rathinam and R. Sengupta, "Matroid intersection and its application to a multiple depot, multiple tsp," 2006.

[17] Z. Friggstad, "Multiple traveling salesmen in asymmetric metrics," in *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*. Springer, 2013.

[18] S. N. Parragh, K. F. Doerner, and R. F. Hartl, "A survey on pickup and delivery models part ii: Transportation between pickup and delivery locations," *Journal für Betriebswirtschaft*, vol. 58, no. 2, 2008.

[19] B. Coltin, "Multi-agent pickup and delivery planning with transfers," Ph.D. dissertation, Carnegie Mellon University, CMU-RI-TR-14-05, 2014.

[20] J. K. Lenstra and A. Kan, "Complexity of vehicle routing and scheduling problems," *Networks*, vol. 11, no. 2, 1981.

[21] M. W. Savelsbergh and M. Sol, "The general pickup and delivery problem," *Transportation science*, vol. 29, no. 1, 1995.

[22] K. Treleaven, M. Pavone, and E. Frazzoli, "Asymptotically optimal algorithms for one-to-one pickup and delivery problems with applications to transportation systems," *IEEE Transactions on Automatic Control*, vol. 59, no. 9, 2013.

[23] P. Caricato, G. Ghiani, A. Grieco, and E. Guerriero, "Parallel tabu search for a pickup and delivery problem under track contention," *Parallel Computing*, vol. 29, no. 5, 2003.

[24] G. Wilfong, "Motion planning in the presence of movable obstacles," *Annals of Mathematics and Artificial Intelligence*, vol. 3, no. 1, 1991.

[25] J. Van Den Berg, M. Stilman, J. Kuffner, M. Lin, and D. Manocha, "Path planning among movable obstacles: a probabilistically complete approach," in *WAFR*, 2008.

[26] M. Stilman, J.-U. Schamburek, J. Kuffner, and T. Asfour, "Manipulation planning among movable obstacles," in *ICRA*, 2007.

[27] Z. Littlefield, S. Zhu, H. Kourtev, Z. Psarakis, R. Shome, A. Kimmel, A. Dobson, A. F. De Souza, and K. E. Bekris, "Evaluating end-effector modalities for warehouse picking: A vacuum gripper vs a 3-finger underactuated hand," in *2016 IEEE International Conference on Automation Science and Engineering (CASE)*. IEEE, 2016, pp. 1190–1195.

[28] O. Ben-Shahar and E. Rivlin, "Practical pushing planning for rearrangement tasks," *IEEE Transactions on Robotics and Automation*, vol. 14, no. 4, 1998.

[29] J. Ota, "Rearrangement of multiple movable objects-integration of global and local planning methodology," in *ICRA*, vol. 2, 2004.

[30] G. Havur, G. Ozbilgin, E. Erdem, and V. Patoglu, "Geometric rearrangement of multiple movable objects on cluttered surfaces," in *ICRA*, 2014.

[31] S. Srivastava, E. Fang, L. Riano, R. Chitnis, S. Russell, and P. Abbeel, "Combined task and motion planning through an extensible planner-independent interface layer," in *ICRA*, 2014.

[32] R. H. Wilson and J.-C. Latombe, "Geometric Reasoning about Mechanical Assembly," *Artificial Intelligence Journal*, vol. 71, no. 2, 1994.

[33] D. Halperin, J.-C. Latombe, and R. H. Wilson, "A General Framework for Assembly Planning: the Motion Space Approach," *Algorithmica*, vol. 26, no. 3-4, 2000.

[34] D. Berenson, S. Srinivasa, and J. Kuffner, "Task space regions: A framework for pose-constrained manipulation planning," *IJRR*, vol. 30, no. 12, 2011.

[35] B. Cohen, S. Chitta, and M. Likhachev, "Single-and dual-arm motion planning with heuristic search," *IJRR*, vol. 33, no. 2, 2014.

[36] A. Krontiris and K. E. Bekris, "Dealing with difficult instances of object rearrangement." in *Robotics: Science and Systems*, 2015.

[37] ——, "Efficiently solving general rearrangement tasks: A fast extension primitive for an incremental sampling-based planner," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2016, pp. 3924–3931.

[38] W. Vega-Brown and N. Roy, "Asymptotically optimal planning under piecewise-analytic constraints," in *WAFR*, 2016.

[39] P. S. Schmitt, W. Neubauer, W. Feiten, K. M. Wurm, G. V. Wichert, and W. Burgard, "Optimal, sampling-based manipulation planning," in *ICRA*. IEEE, 2017.

[40] I. Gurobi Optimization, "Gurobi optimizer reference manual," 2016.

[41] G. Dantzig, R. Fulkerson, and S. Johnson, "Solution of a large-scale traveling-salesman problem," *Journal of the operations research society of America*, vol. 2, no. 4, pp. 393–410, 1954.

[42] L. A. Santaló, *Integral geometry and geometric probability*. Cambridge University Press, 2004.

[43] M. Ajtai, J. Komlós, and G. Tusnády, "On optimal matchings," *Combinatorica*, vol. 4, no. 4, 1984.

[44] J. Yu, S.-J. Chung, and P. G. Voulgaris, "Target assignment in robotic networks: Distance optimality guarantees and hierarchical strategies," *IEEE Transactions on Automatic Control*, vol. 60, no. 2, 2015.

[45] Z. Galil, S. Micali, and H. Gabow, "An O(EV log V) algorithm for finding a maximal weighted matching in general graphs," *SIAM Journal on Computing*, vol. 15, no. 1, 1986.

[46] B. Dezső, A. Jüttner, and P. Kovács, "Lemon–an open source c++ graph template library," *Electronic Notes in Theoretical Computer Science*, vol. 264, no. 5, 2011.

[47] D. Kirkpatrick and P. Liu, "Characterizing minimum-length coordinated motions for two discs," *arXiv preprint arXiv:1607.04005*, 2016.

[48] R. Shome and K. E. Bekris, "Improving the scalability of asymptotically optimal motion planning for humanoid dual-arm manipulators," in *Humanoids*, 2017.

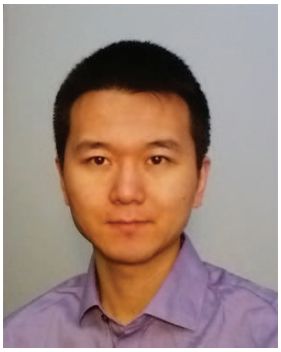
[49] R. Chinta, S. D. Han, and J. Yu, "Coordinating the motion of labeled discs with optimality guarantees under extreme density," in *WAFR*, 2018.

[50] B. Ghosh, "Random distances within a rectangle and between two rectangles," *Bulletin Calcutta Math Soc.*, vol. 43, 1951.

[51] R. Shome and K. E. Bekris, "Anytime multi-arm task and motion planning for pick-and-place of individual objects via handoffs," in *2nd IEEE International Symposium on Multi-Robot and Multi-Agent Systems (MRS)*, New Brunswick, NJ, USA, 08/2019 2019. [Online]. Available: https://arxiv.org/pdf/1905.03179.pdf

Rahul Shome is a Postdoctoral Research Associate at Kavraki Lab, Rice University, working with Prof. Lydia Kavraki. He attained his Ph.D. in Computer Science from Rutgers University where he worked with Prof. Kostas Bekris at the PRACSYS Lab. His research interests include motion planning, manipulation, and task-and-motion planning with a special focus on practical real-world robotic solutions with strong theoretical guarantees.

Kiril Solovey is a Postdoctoral Scholar at the Autonomous Systems Lab, Department of Aeronautics and Astronautics, Stanford University. He received his Ph.D. in Computer Science from Tel Aviv University, Israel. His work focuses on multi-robot systems and their applications to smart mobility, with an emphasis on the design of effective control and decision-making mechanisms. He is the recipient of the Clore Scholars Award, Fulbright Postdoctoral Fellowship, and several best paper awards and nominations.



Jingjin Yu is an Assistant Professor in the Department of Computer Science at Rutgers, the State University of New Jersey. He received his B.S. degree from the University of Science and Technology of China in 1998. He holds M.S. degrees in Chemistry (Univ. Chicago, 2000), Mathematics (Univ. Illinois at Chicago, 2001), and Computer Science (Univ. Illinois at Urbana-Champaign, 2010). He obtained his Ph.D. degree in Electrical and Computer Engineering from the University of Illinois at Urbana-Champaign in 2013,

where he briefly stayed as a postdoctoral researcher. He was a postdoctoral researcher at the Massachusetts Institute of Technology from 2013 to 2015 with a joint appointment at Boston University from 2013 to 2014. He is broadly interested in the areas of robotics and control, focusing on issues related to computational complexity and the design of efficient algorithms with provable guarantees. He is a Siebel Scholar and a recipient of the NSF CAREER Award.

Kostas Bekris is an Associate Professor in the Computer Science department of Rutgers University. He received his MS and PhD degrees in Computer Science from Rice University in 2004 and 2008 respectively. He was an Assistant Professor at the Department of Computer Science and Engineering at the University of Nevada, Reno from 2008 to 2012. He is the recipient of a NASA Early Career Faculty award and his research has been supported by NSF, NASA, DHS and the DoD. His research interests include planning and coordination of robots, especially for systems with many degrees of freedom and significant dynamics, as well as applications to robot manipulation, planetary exploration, cyber-physical systems and physically-realistic virtual agents.

Dan Halperin received his Ph.D. in Computer Science from Tel Aviv University. He then spent three years at the Computer Science Robotics Laboratory at Stanford University. In 1996 he joined the School of Computer Science at Tel Aviv University, where he is currently a full professor. Halperin's main field of research is Computational Geometry and its Applications. A major focus of his work has been in research and development of robust geometric software, principally as part of the CGAL project and library. The application areas he is interested in include robotics, automated manufacturing, algorithmic motion planning and 3D printing. http://acg.cs.tau.ac.il/danhalperin